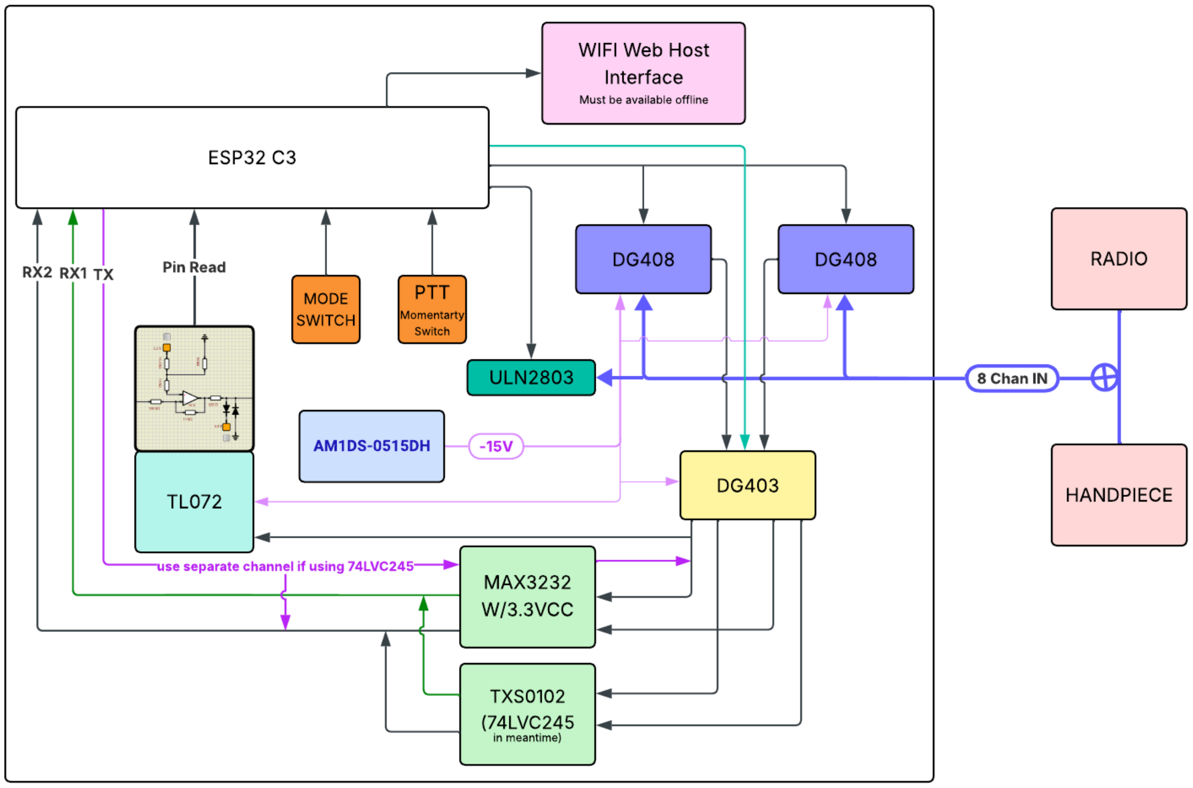
Remote PTT V1

This device is a serial emulating peripheral that sits passively between a UHF radio (RADIO) and its handpiece (HANDPIECE). There are up to 8 active lines (8 Chan IN) that it will detect, read and use to send spoof commands to the radio in order to facilitate remote PTT (Push To Talk) capability. Spoof commands are saved locally on the MCU in non-volatile memory under user defined profiles eg: “Grader 7” or “Hitachi Ex”.

Write an ESP32 sketch for the following & refer to block design. Please ask any questions you require to complete, do not make assumptions and refer to data sheets.

Block Design



Block diagram explained:

- The ESP32 C3 will be referred to as the MCU in this document and is the device's primary control mechanism.
- 8 Chan IN. Each of the 8 lines is unknown and may have on it one of the following:

    - +11~13v
    - GND
    - TTL UART or other serial coms +5v or +3v
    - RS232 +/-12v
- 2 x DG408 IC's will receive the 8 lines each in parallel. They will route serial data in/out and route +/-V & GND for reading. Dual supply +/- 15vDC
- 1 DG403 will route the outputs of the DG408's through the appropriate level shifter.
- TL072 is an opamp with voltage divider and clamps to read each of the 8 lines in safely through the PIN READ pin ADC.
- ULN2803 sinks "learnt" TX pin to GND when "Traditional Handpeice" is selected and PTT button activated. Referred to as "ULN" in code.
- AM1DS-0515DH Dual supply DC DC converter to supply DG408's, DG403 and TL072 with +/-15vDC
- MAX3232 will provide level shifting for and RS232 or 5V UART signals - Referred to as "MAX" in code.
- TXS0102 will provide level shifting for and 3.3 and 5V UART signals - Referred to as "TXS" in code.
- PTT is a momentary switch that is pulled high and activates TX when grounded.
- MODE SWITCH is a SPST switch that selects between LEARN and RUN.
- WIFI Web Host Interface - The MCU will host a website to use as a GUI and will be its primary interface. It will be accessed via the MCU's onboard WIFI.

GUI

The GUI is a simple, attractive and intuitive website that is hosted locally on the MCU so as to be available offline. The GUI will be optimised for use on mobile devices.
When prompting the user to manipulate the transmit button, a large and clear notification needs to be shown on screen.
Working from the top, down:
Always: The state of the MODE SWITCH will be displayed large and bold and the page will change depending on whether LEARN or RUN is selected.

LEARN:

1. A drop down box titled "Profile Name" will display the current profile (blank if none).
2. It will have an add button "+" next to it to add a new profile. Duplicates will not be allowed. User will be notified that name already exists.
3. If a new profile is selected or learn procedure failed, the page will show a large "LEARN" button. If the profile has current settings stored, the large button will say "Re-Learn".
4. Upon the user clicking Learn or Re-Learn, the user will be prompted with 2 options: "Remote Handpiece" Or "Traditional Handpiece"
5. A box will open that displays the serial register. All references to SP will display here.

6. The user will be guided through the LEARN function of the device.
7. If successful, the user will then be notified that the “Learn Procedure” has been successful, be prompted to switch the device to RUN and notified that WIFI connection to the device is no longer required unless changing/adding profiles.
8. The page will return to 3.

RUN

1. A drop down box titled “Current Profile” will display the current profile.
2. If the profile has had a successful calibration and its settings are stored on device, a large green “Status” box will say “RUNNING”. All pins stored under the current profile will be displayed in a list. The function showPins(); (detailed below) can be called for this. A Serial Register will also be opened and display any serial.print() from the code.
3. If the profile has NOT had a successful calibration and its settings are NOT stored on device, a large RED “Status” box will say “LEARNING REQUIRED” with sub text prompting user to change MODE SWITCH to LEARN and complete Learn Procedure.

Coding requirements

Simple and efficient. Commented out **thoroughly** so that a hobbyist level programer can understand **fully** and edit. Split into chunks and named as to the chunks function or relevance to this document.

The DG408’s will be referred to and called as MUX1 & MUX2 in the code. They use a binary address control method. A function should be used to simplify this and call the MUX and its PIN with pins named 1 - 8. Eg: mux(1,5); would call MUX1 and select pin 4 (pins being 0-7 at the hardware level but 1-8 in the code). Within the same function “on” and “off” will be called to enable or disable the mux. Eg: mux(2,on);

In this document, SP: “” means serial print “”.
SP (red): “” make the font colour **red** in the GUI.

A function showPins(); will be created and when called, will SP all pins and their info stored under the current profile. This function is to allow the GUI to call it.

Code Function

On boot it will read the state of MODE SWITCH.

If MODE SWITCH is set to LEARN:

1. Set DG403 input to MAX (Route its outputs to the MAX3232)
2. Awaits input from the GUI
3. If "Remote Handpiece" option is selected by the user in the GUI, store "remote" to non volatile memory as the "Profile Type" under the current profile and :

    3.1. MUX1 is enabled.

    3.2. Each one of MUX1's input pins are driven sequentially through the PIN READ pin ADC.
    Each pin is defined and stored to non volatile memory as:
    pinInfo {
    Definition; //+12v(X), GND(X), Data(X) or Undefined(X) (X starting at 1)
    Mux; // mux1 or mux2
    pinNumber; // 1~8
    ADC; // mapped ADC reading displayed as actual volts

    Pins with **<u>Stable</u>** voltage >11V will be defined as "+12v-(X)" - SP(green): "Power pin (X) (voltage) stored"
    Pins with **<u>Stable</u>** -.5~.5V will be defined as "GND(X)" - SP(green): "GND pin (X) stored"
    Pins with 2.5~6V will be defined as "Data(X)" - SP(green): "Data pin (X) stored"
    Pins with <-2.5V will be defined as "RS232-(X)" - SP(green): "RS232 pin (X) stored"
    Any pins not fitting above constraints will be defined as "Undefined(X) (voltage)" - SP(red): "Undefined pin (X) - (ADC) stored"

    3.3. If any combination of DATA and RS232 pins other than 2 DATA <u>OR</u> 2 RS232 pins are stored, SP(red): "Data pins unclear". GUI returns to "LEARN" step 3 after 5 seconds.

    3.4. If 2 DATA OR 2 RS232 pins are stored, the device will determine which of the data pins is TX & RX by determining which pin transmits first. The commands are separated by less than 200us so it has to be accurate!
    If pins are RS232, keep DG403 set to MAX
    Else if pins are DATA, set DG403 to TXS
    Establish UART RX1 and RX2 at highest reliable baud rate
    Prompt user to "Press & Hold Transmit Button"

Determine which UART transmitted first.
Prompt user to “Release Transmit Button”
Confirm which UART transmitted first.
If not confirmed SP(red): “TX Unclear” and repeat.
Else if confirmed SP(green): “TX Confirmed” and store the TX pin as “TXpin” and the RX pin as “RXpin” under the current profile.
Store DG403 current input state as “DG403_state” under the current profile.

3.5. The device now needs to record and store to non volatile memory under the current profile, the “TXstart” and “TXstop” commands for spoofing.

Prompt the user to “Press & Hold Transmit Button”
If UART received, record the incoming UART command and stop when 32 unchanged bits are received. SP(green): “TXstart Received"
Store as “TXstart”
If UART not received, repeat step 5 times. If still not received, prompt user to contact support.

Prompt user to “Release Transmit Button”
If UART received, record the incoming UART command and stop when 32 unchanged bits are received. SP(green): “TXstop Received"
Store as as “TXstop”
If UART not received, go back to start and repeat step 5 times. If still not received, prompt user to contact support.

Clip both commands to 1 bit after the last low bit and store to non volatile memory under the current profile.
SP(green): “TX Commands Recorded Successfuly!”
Now refer to GUI, LEARN step 7.

4. If “Traditional Handpiece” option is selected by the user in the GUI, store “trad” to non volatile memory as the “Profile Type” under the current profile:

4.1. MUX1 is enabled.

4.2. Each one of MUX1’s input pins are driven sequentially through the PIN READ pin ADC.
Each pin is defined and submitted to non volatile memory as:
pinInfo {
Definition; //+12v(X), GND(X), Data(X) or Undefined(X) (X starting at 1)
Mux; // mux1 or mux2
pinNumber; // 1~8
ADC; // mapped ADC reading displayed as actual volts

Pins with **Stable** voltage >11V will be defined as “+12v-(X)” - SP(green): “Power pin (X) (voltage) stored”

Pins with **Stable** -.5~.5V will be defined as “GND(X)” - SP(green): “GND pin (X) stored”
Pins with 2.5~6V will be defined as “Data(X)” - SP(green): “Data pin (X) stored”
Pins with <-2.5V will be defined as “RS232-(X)” - SP(green): “RS232 pin (X) stored”
Any pins not fitting above constraints will be defined as “Undefined(X) (voltage)” - SP(red): “Undefined pin (X) - (ADC) stored”

4.3. Prompt the user to “Press & Hold Transmit Button”
Each “+12v” pin is driven sequentially through the PIN READ pin ADC and checked for GND.
If a +12v pin reads GND or close to when the transmit button is pressed it is stored as “TXpin(X)” under the current profile.
Prompt user to “Release Transmit Button”
If no change is detected on any pin, repeat 4.3. 5 times. Else prompt user to contact support.
If >1 TXpin is detected SP(red): “TXpin Unclear” and prompt user to contact support.

If MODE SWITCH is set to RUN:

1. Set DG403 input to MAX
2. Use the ADC to sequentially test each pin and confirm its stored ADC reading.
3. If any mismatches > +/-1v, SP(red): (List of any pins that do not match their stored ADC readings by +/- 1v) and Prompt user to select correct profile or Re-Learn.
4. If no mismatches are detected, change DG403 to DG403_state of current profile.
5. Assign interrupt to PTT
6. If current profile type is “trad” assign TXpin to ULN so that when PTT button is pressed, TXpin is sunk to GND through the ULN until it is released. SP(green): “TX Sent”.
7. Else if current profile type is “remote”, transmit “TXstart” on RXpin when PTT activated and transmit “TXstop” on RXpin when PTT released. SP(green): “TX Sent”. (if RS232 mux1 will be used, if DATA use mux2) // check pins in code because we haven't specified this to the AI and also check routing of UART when using 74LVC245//